（一） 概要设计的任务与步骤

1、总体设计的必要性：可以站在全局角度上，花较少成本，从抽象的层次上分析对比多种可能性的系统实现方案和软件结构，从中选出最佳方案和最合理的软件结构，从而用较低成本开发出较高质量的软件系统。

2、总体设计的两个阶段：

（1）系统设计阶段：确定系统的具体实现方案

（2）结构设计阶段：确定软件结构。

3、总体设计的9个步骤：

（1）设想供选择的方案

（2）选取合理的方案

（3）推荐最佳方案

（4）功能分解

（5）设计软件结构

（6）设计数据库

（7）制定测试计划

（8）书写文档

（9）审查和复审

（二） 软件设计的基本原则、抽象与逐步求精方法

传统软件工程方法学采用结构化设计方法（SD）

1、从工程管理角度结构化设计分为两步：

①概要设计：讲软件需求转化为数据结构和软件系统结构

②详细设计：过程设计，通过对结构细化，得到软件详细数据结构和算法

2、设计原理

（1）模块化：把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个整体，可以完成指定的功能满足用户的需求。

①模块定义：又称“构件”一般指用一个名字调用的相邻程序元素序列。

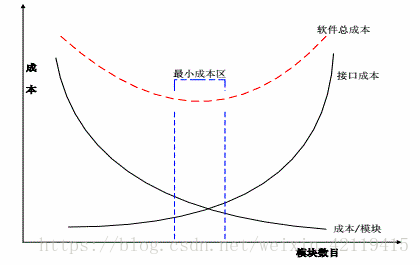
②模块化设计：按适当的原则把软件划分为一个个较小的、相关而又相对独立的模块。

但无线的划分模块会导致接口成本提高。

————————————————

版权声明：本文为CSDN博主「文文的博客」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_42119415/article/details/82901615



2）抽象：抽出事物的本质特性，暂不考虑细节

（3）求精：能集中精力解决主要问题，尽量推迟对细节问题的考虑，实际上是一个细化的过程与抽象是互补的概念

（4）信息隐藏：每个模块的实现细节对于其他模块来说是隐藏的。模块中所有包含的信息是不允许其他不需要这些信息的模块访问的。每个客户只能通过接口来了解该模块，而所有的实现都隐藏起来的。

（5）局部化：把一些关系密切的软件元素物理地址放的彼此靠近

（6）模块独立：是模块化、抽象、信息隐藏和局部化概念的直接结果。具有独立功能且和其他模块没有过多作用

为什么模块独立？两条理由：容易分工合作；容易测试和维护，修改工作量比较小，错误传播范围小，扩充功能容易。

3、两个定性度量标准：耦合、内聚

（1）耦合：软件结构中间不同模块间互联程度度量

取决：模块接口复杂程度，通过接口数据。追求尽可能松散耦合系统。

耦合的三大类：无耦合、松散耦合、紧密耦合（避免）

常见：

非直接耦合：两模块分别能独立地工作不需要另一个模块存在

数据耦合：两模块通过参数交换数据信息

控制耦合：两个模块通过参数交换控制信息（包括数字形式）。如：依赖控制信息决定执行步骤

公共耦合：

两种可能：①（松散）一模块送数据，一模块取数据，等价数据

②（紧密）两个模块既往公共环境送又从里面取，介于数据耦合和控制耦合之间

内容耦合（相当紧密）：

①一模块访问另一模块内部数据

②一模块不通过正常入口转到另一模块内部

③两模块有部分程序代码重叠（汇编程序）

④一模块有多个入口

原则：尽量使用数据耦合，少使用控制耦合，限制使用公共环境耦合，完全不用内容耦合。

（2）内聚：模块内各元素彼此结合紧密程度

功能内聚（10分）：一模块中各部分是完成某一功能必不可少的组成部分

顺序内聚（9分）：模块内处理元素功能紧密相关，顺序执行。如：修改学生信息，先查找后修改

通信内聚（中等7分）：一模块内各功能部分都使用相同输入数据，或产生相同输出数据。如：根据编号获得配件单价和库存量输出“库存量”、“单价”

过程内聚（中等5分）：模块内处理元素相关，特定次序执行。如：把流程图中循环部分、判定部分，计算部分分成三个模块，这三个模块内聚为过程内聚

时间内聚（3分尽量不出现）：多为多个功能模块，要求所有功能在同一时间内执行。如：初始化模块

逻辑内聚（1分尽量不出现）：一模块完成功能在逻辑上属相同类似一类。

偶然内聚（0分尽量不出现）：模块内各部分没有联系，即使有也很松散

4、启发规则

（1）改进软件结构提高模块独立性

（2）模块规模应适中：通常语句行数在50~100行（一页纸），最多500行

（3）深度、宽度、扇出和扇入都应适当

深度：软件结构控制层数，标志一系统大小和复杂程度

宽度：软件结构同一层模块最大值，越大系统越复杂

扇出：一模块直接控制（调用）模块数3~9

扇出：有多少上级模块直接调用它，越大共享该模块上级模块越多

（4）模块作用域应在控制域内

作用域：受该模块内判定影响的所有模块

控制域：模块本身及所有直接或间接从属它的模块集合

改善一：判定点上移

改善二：将在作用域不在控制域内的模块下移

（5）降低模块接口复杂程度

（6）设计单接口，单出口模块

（7）模块功能可预测：输入数据相同，产生同样输出；模块功能防止过分受限。

（三） 详细设计的任务

1、任务：确定模块算法；确定模块使用数据结构；确定接口（系统外部接口、用户界面、内部模块间接口细节、输入数据和输出数据）

2、人机界面设计

①系统响应时间：长度0.1~1秒正常；处理1-10秒鼠标显示成沙漏；处理10~18秒由为帮助显示成处理进度；18秒以上显示处理窗口或显示进度条

②用户帮助措施：手册和联机帮助两种

③出错信息处理：以用户可以理解的术语；提供清楚，易理解报错信息；从错误中恢复的建设性意见；可造成负面后果。

④命令交互：建议保留命令形式交互方式（控制序列、功能键、控制宏机制、键入命令）

3、程序流程图

优点：对控制流出层的描述很直观，便于初学者

缺点：①程序流程图本质上不是逐步求精的好工具，它诱使程序员过早的考虑程序的控制流程，而不考虑程序的全局结构

②程序流程图中箭头代表控制流，因此程序有不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制

③程序流程图不易表示数据结构

4、盒图（N-S图）

特点：

①功能域明确

②不可能任意转移控制

③很容易确定局部和全程数据的作用域

④很容易表现嵌套关系，也可以表示模块的层次

5、PAD图

特点：

①设计出的程序必然是结构化程序

②描绘的程序结构十分清晰

③表示程序逻辑，易读、易懂、易记

④容易将PAD图转换成高级语言源程序

⑤可用于表示程序逻辑，也可用于描绘数据结构

⑥支持自顶向下、逐步求精方法的使用

6、判定表：能够清晰表示复杂条件组合与应做动作间对应关系

四部分

左上：列出所有条件

左下：所有可能的动作

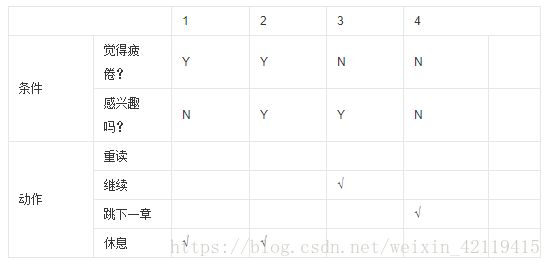
右上：表示各种条件组合的矩阵

右下：和每种条件组合相对应的动作

————————————————

版权声明：本文为CSDN博主「文文的博客」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/



weixin\_42119415/article/details/82901615

7、判定树

优点：形式简单，易看出含义，易于掌握和使用

缺点：简介性不如判定表，相同数据重复写多遍，越接近叶端重复次数越多

8、PDL：伪码，用正文形式表示数据和处理过程设计工具

PDL具有严格关键字外部语法，定义控制结构和数据结构

PDL表示实际操作和条件的内部语法灵活自由，适应各种工程项目需要。

9、程序复杂度（小于等于10）

使用比较广泛的cCabe方法

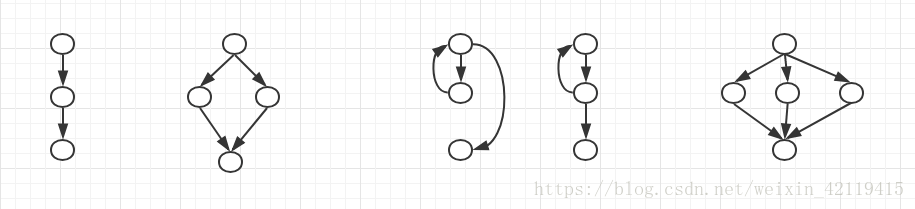
根据过程设计结果画出相应的流图

流图描述程序控制流，基本图形符号

————————————————

版权声明：本文为CSDN博主「文文的博客」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_42119415/article/details/82901615



计算环形复杂度：

三种方法：V(G)=区域数；V(G)=E-N+2（E为流图边数，N为流图节点数）；V(G)=P+1（P为判定点数）

（四） 结构化程序设计的概念和思想

1、结构化程序设计

（1）经典定义：如果一个程序的代码块仅仅通过顺序、选择、和循环3种基本控制结构进行连接，并每个代码块只有一个入口和一个出口，则称这个程序是结构化的。

（2）扩展定义：可限制使用GOTO语句，DO\_UNTIL、DO\_CASE

（3）修正定义：leave和break，可从循环中转移出来。

（五） 面向对象程序设计的概念和思想

数据结构既是影响程序的结构也是影响程序处理过程，可以数据结构导出程序的处理过程，适合详细设计

两种面向数据结构设计方法：Jackson和Warnier方法

1、Jackson图

优点：

①便于表示层次结构，是对结构进行自顶向下分解的有力工具

②形象直观可读性好

③既能表示数据结构也能表示程序结构

步骤：

①确定输入数据和输出数据逻辑结构，用Jackson图表达

②确定输入结构和输入结构中有对应关系（因果）的单元

③描绘数据结构的Jason图导出描绘程序结构的Jason图

④列出所有操作和条件，分配到Jason图中

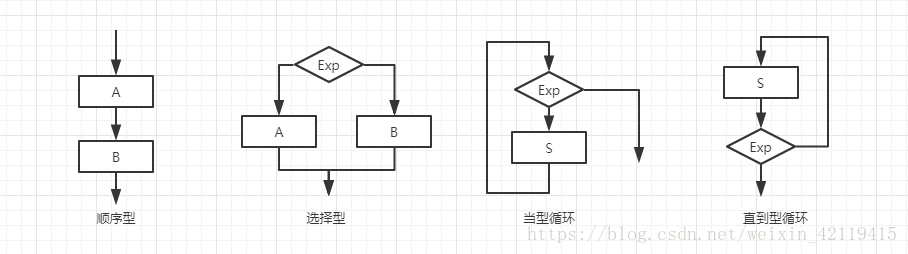
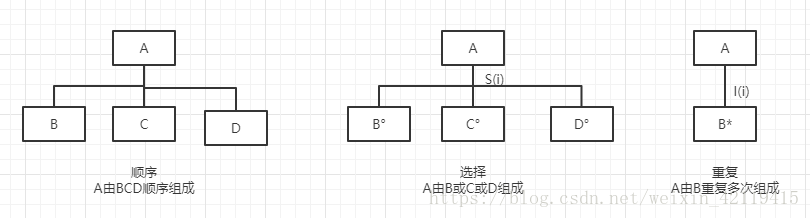
⑤用伪码表示

描述数据结构图形符号：顺序、选择、重复

————————————————

版权声明：本文为CSDN博主「文文的博客」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_42119415/article/details/82901615



（七） 模型-视图-控制器框架（MVC）

本节参考《百度百科》

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。

　　通常模型对象负责在数据库中存取数据。

View（视图）是应用程序中处理数据显示的部分。

　　通常视图是依据模型数据创建的。

Controller（控制器）是应用程序中处理用户交互的部分。

　　通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

优点

耦合性低

视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

模型是自包含的，并且与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。如果把数据库从MySQL移植到Oracle，或者改变基于RDBMS数据源到LADP，只需改变模型即可。一旦正确的实现了模型，不管数据来自数据库或是LDAP服务器，视图将会正确的显示它们。由于运用MVC的应用程序的三个部件是相互独立，改变其中一个不会影响其它两个，所以依据这种设计思想能构造良好的松耦合的构件。

重用性高

随着技术的不断进步，需要用越来越多的方式来访问应用程序。MVC模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何WEB（HTTP）浏览器或者无线浏览器（wap），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。例如，很多数据可能用HTML来表示，但是也有可能用WAP来表示，而这些表示所需要的命令是改变视图层的实现方式，而控制层和模型层无需做任何改变。由于已经将数据和业务规则从表示层分开，所以可以最大化的重用代码了。模型也有状态管理和数据持久性处理的功能，例如，基于会话的购物车和电子商务过程也能被Flash网站或者无线联网的应用程序所重用。

生命周期成本低

MVC使开发和维护用户接口的技术含量降低。

部署快

使用MVC模式使开发时间得到相当大的缩减，它使程序员（Java开发人员）集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

可维护性高

分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。

有利软件工程化管理

由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化管理程序代码。控制器也提供了一个好处，就是可以使用控制器来联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

缺点

没有明确的定义

完全理解MVC并不是很容易。使用MVC需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。同时由于模型和视图要严格的分离，这样也给调试应用程序带来了一定的困难。每个构件在使用之前都需要经过彻底的测试。

不适合小型，中等规模的应用程序

花费大量时间将MVC应用到规模并不是很大的应用程序通常会得不偿失。

增加系统结构和实现的复杂性

对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。

视图与控制器间的过于紧密的连接

视图与控制器是相互分离，但却是联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。

视图对模型数据的低效率访问

依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

一般高级的界面工具或构造器不支持模式

改造这些工具以适应MVC需要和建立分离的部件的代价是很高的，会造成MVC使用的困难

————————————————

版权声明：本文为CSDN博主「文文的博客」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_42119415/article/details/82901615